MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2022/2023 (TERM ID 2210)

ECE2216 – MICROCONTROLLER AND MICROPROCESSOR SYSTEMS

(All sections / Groups)

15 FEBRUARY 2023
2.30 p.m - 4.30 p.m
(2 hour)

INSTRUCTIONS TO STUDENTS

1. This question paper consists of 14 pages with 4 questions only.
2. Answer **ALL** questions. All questions carry equal marks and the distribution of the marks for each question is given.
3. Please write all your answers in your own Answer Booklet.

Question 1

(a)

i. Convert the octal number $\mathbf{7542_8}$ to binary bits and then hexadecimal digits.

[4 marks]

ii. Determine the range of a 5-bit 2's complement number, *X*.

[2 marks]

(b)

i. Draw the diagrams to differentiate between microcontroller and microprocessor.

[4 marks]

ii. Highlight **ONE** application perspective to distinguish microcontrollers from microprocessors.

[2 marks]

(c) An 8-bit 8051 microcontroller needs to access 8-Kbyte of data in a memory. Each memory location can store 8-bit data.

i. What is the size of the data bus required?

[2 marks]

ii. What is the size of the address bus required? Show the computation steps.

[6 marks]

(d) What is the information on the address and data buses during an opcode fetch? What is the direction of information flow on these buses during an opcode fetch?

[5 marks]

Continued....

Question 2

(a) State the 8051 control bus signals that are used to select external EPROMs and external RAMs.

[3 marks]

(b) What instruction sequences that can be used to copy the content of R7 to external RAM location 100H?

[6 marks]

(c) (i) What is the purpose of address decoding? Justify your answer.

[3 marks]

(ii) Design an 8051-based memory system using 8-Kbytes ROM and 8-Kbytes RAM external ICs to form 24-Kbytes of ROM and 24-Kbytes of RAM. Using this external ICs and decoder chip draw the configuration of the system showing the 8051 signal lines to be used for address, data & control buses.

[10 marks]

(d) What is the state of the Parity (P) bit in the Program Status Word (PSW) after execution of each of the following instructions?

(i) MOV A, # 55H
(ii) MOV A, #0F8H
(iii) MOV A, # 0FFH

[3 marks]

Continued....

Question 3

(a) The incomplete instruction sequence and the machine codes for the 8051 microcontroller are shown in Table Q3(a).

Table Q3(a)

Address	Instruction	Machine codes
0000H	MOV P1, #0FFH	
0003H		75H A0 FFH
0006H	CHK: ANL P1, #5AH	
0009H	SJMP CHK	

i. Complete and reproduce the table in your answer script by converting each instruction to the corresponding machine codes and converting the machine codes to the corresponding assembly instructions.

[4 marks]

ii. Identify the addressing mode of the source and destination operands for the first and addressing mode for the fourth instruction.

[2 marks]

iii. Determine the total size of the program in bytes. [2 marks]

Hint: Refer to the opcode map in the Appendix. Address of P1 = 90H, Address of P2 = A0H.

(b) An embedded system engineer has developed an assembly program for the 8051 microcontroller as shown in Figure Q3(b). Initial values: [55H] = 03H, [A003H] = 05H, [A005H] = 07H.

```
        ORG  0000H
        MOV  R1, 55H
        MOV  A, R1
        MOV  DPTR, #0A00H
        MOVC A, @A+DPTR
        MOVC A, @A+DPTR
        MOV  R1, A
        MOV  R0, #30H
        MOV  R3, #2
LOOP:   MOV A, R1
        ADD A, RO
        MOV @R0, A
        INC R0
        DJNZ R3, LOOP
        SJMP $  ; program stops here
        END
```

Figure Q3(b)

Continued....

You are required to perform the following tasks for the program listed in Figure Q3(b):

i. Analyze the program to determine the address of the affected memory locations in the internal RAM and their final contents after the program has been executed. [3 marks]

ii. Determine the final contents of registers R0 and A (Accumulator) when the program has stopped. [3 marks]

iii. Calculate the address represented by the label "LOOP". [2 marks]

Hint: Refer to the opcode map in the Appendix.

(c) Complete the instruction sequence shown in Figure Q3(c) to perform the following arithmetic operation and store the final result in register R3.

$$(98_{10} - 66_{10}) \times (25_{10} + 22_{10})$$

```
        CLR __
        MOV A, #98
        ____ A, #___
        MOV B, A
        MOV A, #25
        ___ A, #22
        MUL __
        MOV __, A
```

Figure Q3(c)

[3 marks]

Continued....

(d) Complete an assembly program in Figure Q3(d) for the 8051 microcontroller to generate a 100 kHz square wave (70% duty cycle) at pin P2.0 by using Timer 1 operating in Mode 2. Assume that the microcontroller is operating with a 12 MHz crystal oscillator.

```
;The period of the square wave, T = ___
;For 70% duty cycle, ___% of T is high = ___µs
; ___% of T is low = ___µs
;8051 time per instruction is 12MHz/12 = 1µs
;TH1 high = _________________
;TH1 low = _________________

            ORG   0000H
            MOV TMOD,#___
HERE:       MOV TH1,#___
            SETB P2.0
            ACALL DELAY1
            MOV TH1,#____
            CLR ____
            ACALL DELAY1
            SJMP HERE
DELAY1:     SETB ___
AGAIN1:     JNB TF1,AGAIN1
            CLR TR1
            CLR TF1
            RET
         END
```

Figure Q3(d)

[6 marks]

Question 4:

(a) An 8051 microcontroller is equipped with an on-chip serial port for communication with external devices using UART.

i. Complete the value for SCON, TMOD, TH1 and statements in the process boxes and decision box for the program flow in Figure Q4(a). It is used to receive bytes of data serially at 2400 baud rate, and put them in P2, continuously with crystal frequency of 11.0592MHz using serial port Mode 1. Odd parity is expected in the 8th bit and the carry flag will be set if there is a parity error.

[4 marks]

ii. Complete an instruction sequence in Figure Q4(b) to implement the program flow in Question 4(a)(i).

Continued....

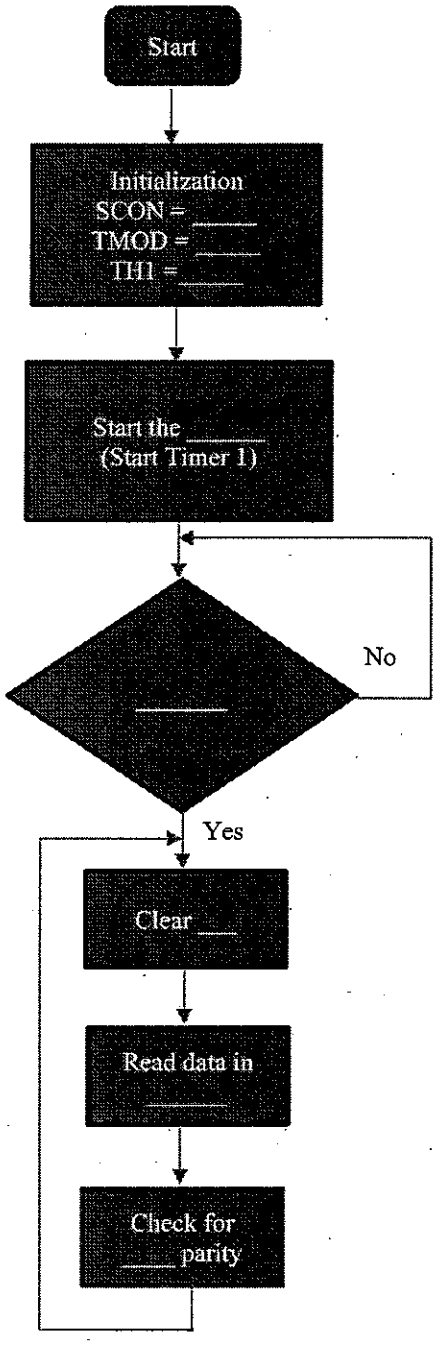


Figure Q4(a)

Continued....

```
        ORG   0000H
        MOV   SCON, #___   ; serial port mode 1
        MOV   TMOD, #___   ; Timer 1, mode 2
        MOV   TH1, #___    ; set baud rate at 2400
        ANL   87H, #01111111B   ; do not double the baud rate
        SETB  ____         ; start Timer 1
HERE:   JNB   RI, HERE     ; wait until character is received
        CLR   RI           ; prepare for next character
        MOV   A,___        ; move received data in ACC
        MOV   C,P          ; obtain parity value
        ___   C            ; set C flag if P is not set
        CLR   ACC.7        ; remove parity bit from data
        MOV   P2, A        ; send data to port 2
        SJMP  HERE         ; continue to receive character
        END
```

Figure Q4(b)

[3 marks]

iii. Complete a subroutine in Figure Q4(c) called "Msg_Ack" to transmit the string "Ack" serially via UART.

```
Msg_Ack:
        MOV ____, #'A'
        JNB __, $
        CLR __
        MOV ____, #'c'
        ___ __, $
        CLR __
        MOV ____, #'k'
        JNB __, $
        CLR __
        RET
```

Figure Q4(c)

[5 marks]

(b) As an intern in your company, you need to analyse an 8051 assembly code shown in Figure Q4(d). Determine the type of interrupt that have been enabled and hence the functionality of the code.

(Hints: The microcontroller is operating with an 11.0592MHz crystal oscillator. Assume active low connection for both LED and active low connection for the switch.)

Continued....

```
        ORG     0000H
        LJMP    main
        ORG     001BH
        LJMP    T1ISR

        ORG     0030H
main:   SETB    P2.2            ;Configured as
                                input
        MOV     TMOD,#10H       ;configure timer1
        MOV     TH1,#04BH
        MOV     TL1,#0FDH       ;50ms
        MOV     IE, #088H
        SETB    TR1

loop:   SETB    P1.1            ;turn off LED2
        JB      P2.2,$
        CLR     P1.1            ;turn on LED2
        MOV     A,#20
        ACALL   Delay
        SJMP    loop

T1ISR:  CLR     TR1
        MOV     TH1,#4BH
        MOV     TL1,#0FDH
        CPL     P1.0            ;toggle LED1
        SETB    TR1
        RETI

Delay:  MOV     R0,A
Repeat: MOV     R1,#10
Again:  MOV     R2,#10
Here:   NOP
        NOP
        DJNZ    R2,Here
        DJNZ    R1,Again
        DJNZ    R0,Repeat
        RET
```

Figure Q4(d)

[5 marks]

Continued....

(c) The following program generates a saw tooth signal by using Digital-to-Analogue (DAC) converter. The DAC is connected to port-1 of an 8051-microcontroller operating with a 12 MHz crystal.

```
        STEP  EQU 1
MAIN:   MOV   P1, A
        ADD   A, #STEP
        SJMP  MAIN
```

i. What is the frequency of the saw tooth signal? [5 marks]

ii. What value for STEP will achieve an output frequency of 10KHz? [3 marks]

End of Paper

APPENDIX

Special Function Register Formats

Interrupt Enable (IE)

Bit Addr.	AFH	-	-	ACH	ABH	AAH	A9H	A8H
Name	**EA**	**-**	**-**	**ES**	**ET1**	**EX1**	**ET0**	**EX0**

BIT	SYMBOL	FUNCTION (Enable=1, Disable=0)
IE.7	EA	Global enable/disable. EA = 1, each individual source is enabled/disabled by setting/clearing its enable bit. EA = 0, disable all interrupts.
IE.6	-	Undefined
IE.5	-	Not implemented in 8051. ET2 for 8052.
IE.4	ES	Serial port interrupt enable bit.
IE.3	ET1	Timer 1 interrupt enable bit.
IE.2	EX1	External interrupt enable bit.
IE.1	ET0	Timer 0 interrupt enable bit.
IE.0	EX0	External interrupt enable bit.

Interrupt Priority (IP)

Bit Addr.	-	-	-	BCH	BBH	BAH	B9H	B8H
Name	**-**	**-**	**-**	**PS**	**PT1**	**PX1**	**PT0**	**PX0**

BIT	SYMBOL	FUNCTION (Enable=1, Disable=0)
IP.7	-	Undefined
IP.6	-	Undefined
IP.5	-	Not implemented in 8051. PT2 for 8052.
IP.4	PS	Serial port interrupt priority bit.
IP.3	PT1	Timer 1 interrupt priority bit.
IP.2	PX1	External interrupt priority bit.
IP.1	PT0	Timer 0 interrupt priority bit.
IP.0	PX0	External interrupt priority bit.

Interrupt Vectors

Interrupt Source	Flag	Vector Address
System Reset	RST	0000H
External 0	IE0	0003H
Timer 0	TF0	000BH
External 1	IE1	0013H
Timer 1	TF1	001BH
Serial Port	RI & TI	0023H
Timer 2 (8052)	TF2 or EXF2	002BH

Program Status Word (PSW)

Bit Addr.	D7H	D6H	D5H	D4H	D3H	D2H	-	D0H
Name	**CY**	**AC**	**F0**	**RS1**	**RS0**	**0V**	-	**P**

Serial Control (SCON)

Bit Addr.	9FH	9EH	9DH	9CH	9BH	9AH	99H	98H
Name	**SM0**	**SM1**	**SM2**	**REN**	**TB8**	**RB8**	**TI**	**RI**

BIT	**SYMBOL**	**FUNCTION**
SCON.7	SM0	Serial port mode bit 0 (see Table A.1).
SCON.6	SM1	Serial port mode bit 1 (see Table A.1).
SCON.5	SM2	Serial port mode bit 2; enables multiprocessor communications in modes 2 and 3; RI will not be activated if received 9th bit is 0. In mode 1, if SM2 = 1, then RI will be activated only if a valid stop bit was received. In mode 0, SM2 should be 0.
SCON.4	REN	Receiver enable; must be set to receive characters.
SCON.3	TB8	Transmit bit 8; 9th bit transmitted in modes 2 and 3; set/cleared by software.
SCON.2	RB8	Receive bit 8; 9th bit received.
SCON.1	TI	Transmit interrupt flag; set at end of character transmission; cleared by software.
SCON.0	RI	Receive interrupt flag; set at end of character reception; cleared by software.

Table A.1 The 8051 Serial Port Mode Selection

SM0	**SM1**	**Mode**	**Description**	**Baud Rate**
0	0	0	Shift register	Fixed
0	1	1	8-bit UART	Variable
1	0	2	9-bit UART	Fixed
1	1	3	9-bit UART	Variable

PCON (Power Control)

Bit	7	6	5	4	3	2	1	0
Name	**SMOD**	-	-	-	**GF1**	**GF0**	**PD**	**IDL**

BIT	**SYMBOL**	**FUNCTION**
7	SMOD	Double the serial port baud rate when it is set to 1
3	GF1	General purpose flag bit 1
2	GF0	General purpose flag bit 0
1	PD	Activate the microcontroller power-down mode when it is set to 1
0	IDL	Activate the microcontroller idle mode when it is set to 1

Timer Control (TCON)

Bit Addr.	8FH	8EH	8DH	8CH	8BH	8AH	89H	88H
Name	**TF1**	**TR1**	**TF0**	**TR0**	**IE1**	**IT1**	**IE0**	**IT0**

BIT	SYMBOL	FUNCTION
TCON.7	TF1	Timer-1 overflow flag. Set by hardware on overflow. Cleared by hardware when processor vectors to interrupt routine. Must be cleared by software when not involve interrupt
TCON.6	TR1	Timer-1 run control bit. Set/cleared by software to turn timer/counter on/off.
TCON.5	TF0	Timer-0 overflow flag. Do the same function as TF1 but for Timer-0.
TCON.4	TR0	Timer-0 run control bit. Do the same function as TR1 but for Timer-0.
TCON.3	IE1	External interrupt-1 edge flag. Set by hardware when interrupt-1 falling edge is detected. Cleared by hardware when interrupt is processed.
TCON.2	IT1	Interrupt-1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.
TCON.1	IE0	External interrupt-1 edge flag. Do the same function as IE1 but for external interrupt-0.
TCON.0	IT0	Interrupt-0 Type control bit. Do the same function as IT1 but for external interrupt-0.

Timer Mode (TMOD)

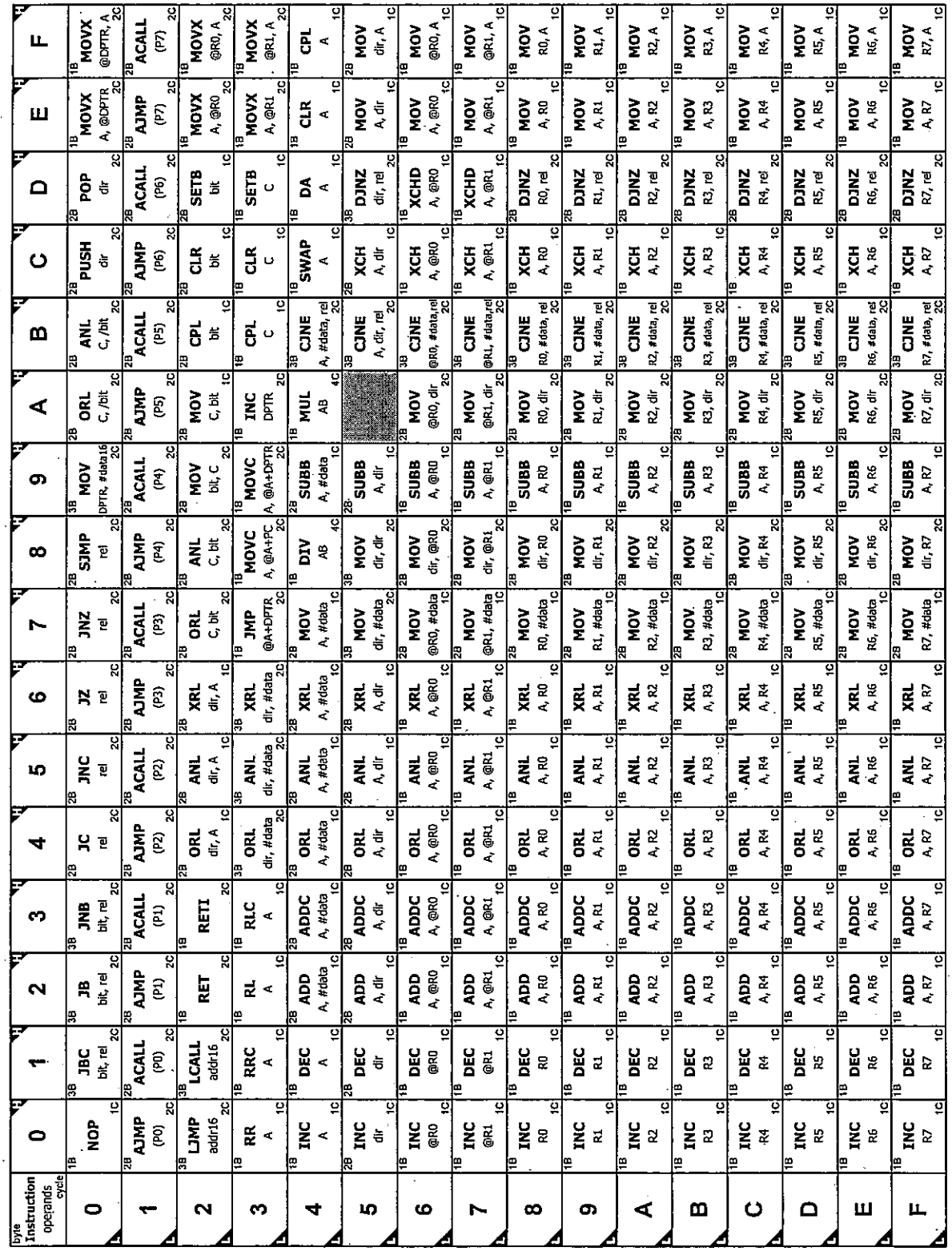
Bit	7	6	5	4	3	2	1	0
Name	**GATE**	**C/T**	**M1**	**M0**	**GATE**	**C/T**	**M1**	**M0**

BIT	SYMBOL	FUNCTION
TMOD.7	GATE1	When this bit is set the timer will only run when INT1 (P3.3) is high (hardware control). When this bit is cleared the timer will run regardless of the state of INT1 (software control).
TMOD.6	C/T1	Timer / Counter select bit. $C / \overline{T} = 0 \rightarrow$ Timer operation. $C / \overline{T} = 1 \rightarrow$ Counter operation.
TMOD.5	M1	Mode selection bits (see Table A.2). [for timer 1]
TMOD.4	M0	Mode selection bits (see Table A.2). [for timer 1]
TMOD.3	GATE0	Exactly the same function as GATE1 but for Timer0
TMOD.2	C/T0	Exactly the same function as C/T1 but for Timer0
TMOD.1	M1	Mode selection bits (see Table A.2). [for timer 0]
TMOD.0	M0	Mode selection bits (see Table A.2). [for timer 0]

Table A.2 Timer Mode Selection

M1	M0	Timer Mode	Description of Mode
0	0	0	13-bit Timer
0	1	1	16-bit Timer
1	0	2	8-bit auto-reload
1	1	3	Split timer mode

Opcode Map


byte
Instruction operands
cycle
0 NOP 1C | JBC bit, rel 3B 2C | JB bit, rel 3B 2C | JNB bit, rel 3B 2C | JC rel 2B 2C | JNC rel 2B 2C | JZ rel 2B 2C | JNZ rel 2B 2C | SJMP rel 2B 2C | MOV DPTR, #data16 3B 2C | ORL C, /bit 2B 2C | ANL C, /bit 2B 2C | PUSH dir 2B 2C | POP dir 2B 2C | MOVX A, @DPTR 1B 2C | MOVX @DPTR, A 1B 2C
1 AJMP (P0) 2B 2C | ACALL (P0) 2B 2C | AJMP (P1) 2B 2C | ACALL (P1) 2B 2C | AJMP (P2) 2B 2C | ACALL (P2) 2B 2C | AJMP (P3) 2B 2C | ACALL (P3) 2B 2C | AJMP (P4) 2B 2C | ACALL (P4) 2B 2C | AJMP (P5) 2B 2C | ACALL (P5) 2B 2C | AJMP (P6) 2B 2C | ACALL (P6) 2B 2C | AJMP (P7) 2B 2C | ACALL (P7) 2B 2C
2 LJMP addr16 3B 2C | LCALL addr16 3B 2C | RET 1B 2C | RETI 1B 2C | ORL dir, A 2B 1C | ORL dir, #data 3B 2C | ANL dir, A 2B 1C | ANL dir, #data 3B 2C | ORL C, bit 2B 2C | JMP @A+DPTR 1B 2C | MOV bit, C 2B 2C | MOV C, bit 2B 1C | CPL bit 2B 1C | SETB bit 2B 1C | MOVX A, @R0 1B 2C | MOVX @R0, A 1B 2C
3 RR A 1B 1C | RRC A 1B 1C | RL A 1B 1C | RLC A 1B 1C | ORL A, #data 2B 1C | ORL dir, A 2B 1C | XRL A, #data 2B 1C | XRL dir, #data 3B 2C | MOVC A, @A+PC 1B 2C | MOVC A, @A+DPTR 1B 2C | INC DPTR 1B 2C | CPL C 1B 1C | CLR C 1B 1C | SETB C 1B 1C | MOVX A, @R1 1B 2C | MOVX @R1, A 1B 2C
4 INC A 1B 1C | DEC A 1B 1C | ADD A, #data 2B 1C | ADDC A, #data 2B 1C | ORL A, #data 2B 1C | ORL dir, #data 3B 2C | ANL A, #data 2B 1C | ANL dir, #data 3B 2C | XRL A, #data 2B 1C | XRL dir, #data 3B 2C | MUL AB 1B 4C | DIV AB 1B 4C | SWAP A 1B 1C | DA A 1B 1C | CLR A 1B 1C | CPL A 1B 1C
5 INC dir 2B 1C | DEC dir 2B 1C | ADD A, dir 2B 1C | ADDC A, dir 2B 1C | ORL A, dir 2B 1C | ANL A, dir 2B 1C | XRL A, dir 2B 1C | MOV dir, #data 3B 2C | SUBB A, dir 2B 1C | MOV dir, dir 3B 2C | CJNE A, dir, rel 3B 2C | CJNE A, #data, rel 3B 2C | XCH A, dir 2B 1C | DJNZ dir, rel 3B 2C | MOV A, dir 2B 1C | MOV dir, A 2B 1C
6 INC @R0 1B 1C | DEC @R0 1B 1C | ADD A, @R0 1B 1C | ADDC A, @R0 1B 1C | ORL A, @R0 1B 1C | ANL A, @R0 1B 1C | XRL A, @R0 1B 1C | MOV @R0, #data 2B 1C | SUBB A, @R0 1B 1C | MOV dir, @R0 2B 2C | MOV @R0, dir 2B 2C | CJNE @R0, #data, rel 3B 2C | XCH A, @R0 1B 1C | XCHD A, @R0 1B 1C | MOV A, @R0 1B 1C | MOV @R0, A 1B 1C
7 INC @R1 1B 1C | DEC @R1 1B 1C | ADD A, @R1 1B 1C | ADDC A, @R1 1B 1C | ORL A, @R1 1B 1C | ANL A, @R1 1B 1C | XRL A, @R1 1B 1C | MOV @R1, #data 2B 1C | SUBB A, @R1 1B 1C | MOV dir, @R1 2B 2C | MOV @R1, dir 2B 2C | CJNE @R1, #data, rel 3B 2C | XCH A, @R1 1B 1C | XCHD A, @R1 1B 1C | MOV A, @R1 1B 1C | MOV @R1, A 1B 1C
8 INC R0 1B 1C | DEC R0 1B 1C | ADD A, R0 1B 1C | ADDC A, R0 1B 1C | ORL A, R0 1B 1C | ANL A, R0 1B 1C | XRL A, R0 1B 1C | MOV R0, #data 2B 1C | SUBB A, R0 1B 1C | MOV dir, R0 2B 2C | MOV R0, dir 2B 2C | CJNE R0, #data, rel 3B 2C | XCH A, R0 1B 1C | DJNZ R0, rel 2B 2C | MOV A, R0 1B 1C | MOV R0, A 1B 1C
9 INC R1 1B 1C | DEC R1 1B 1C | ADD A, R1 1B 1C | ADDC A, R1 1B 1C | ORL A, R1 1B 1C | ANL A, R1 1B 1C | XRL A, R1 1B 1C | MOV R1, #data 2B 1C | SUBB A, R1 1B 1C | MOV dir, R1 2B 2C | MOV R1, dir 2B 2C | CJNE R1, #data, rel 3B 2C | XCH A, R1 1B 1C | DJNZ R1, rel 2B 2C | MOV A, R1 1B 1C | MOV R1, A 1B 1C
A INC R2 1B 1C | DEC R2 1B 1C | ADD A, R2 1B 1C | ADDC A, R2 1B 1C | ORL A, R2 1B 1C | ANL A, R2 1B 1C | XRL A, R2 1B 1C | MOV R2, #data 2B 1C | SUBB A, R2 1B 1C | MOV dir, R2 2B 2C | MOV R2, dir 2B 2C | CJNE R2, #data, rel 3B 2C | XCH A, R2 1B 1C | DJNZ R2, rel 2B 2C | MOV A, R2 1B 1C | MOV R2, A 1B 1C
B INC R3 1B 1C | DEC R3 1B 1C | ADD A, R3 1B 1C | ADDC A, R3 1B 1C | ORL A, R3 1B 1C | ANL A, R3 1B 1C | XRL A, R3 1B 1C | MOV R3, #data 2B 1C | SUBB A, R3 1B 1C | MOV dir, R3 2B 2C | MOV R3, dir 2B 2C | CJNE R3, #data, rel 3B 2C | XCH A, R3 1B 1C | DJNZ R3, rel 2B 2C | MOV A, R3 1B 1C | MOV R3, A 1B 1C
C INC R4 1B 1C | DEC R4 1B 1C | ADD A, R4 1B 1C | ADDC A, R4 1B 1C | ORL A, R4 1B 1C | ANL A, R4 1B 1C | XRL A, R4 1B 1C | MOV R4, #data 2B 1C | SUBB A, R4 1B 1C | MOV dir, R4 2B 2C | MOV R4, dir 2B 2C | CJNE R4, #data, rel 3B 2C | XCH A, R4 1B 1C | DJNZ R4, rel 2B 2C | MOV A, R4 1B 1C | MOV R4, A 1B 1C
D INC R5 1B 1C | DEC R5 1B 1C | ADD A, R5 1B 1C | ADDC A, R5 1B 1C | ORL A, R5 1B 1C | ANL A, R5 1B 1C | XRL A, R5 1B 1C | MOV R5, #data 2B 1C | SUBB A, R5 1B 1C | MOV dir, R5 2B 2C | MOV R5, dir 2B 2C | CJNE R5, #data, rel 3B 2C | XCH A, R5 1B 1C | DJNZ R5, rel 2B 2C | MOV A, R5 1B 1C | MOV R5, A 1B 1C
E INC R6 1B 1C | DEC R6 1B 1C | ADD A, R6 1B 1C | ADDC A, R6 1B 1C | ORL A, R6 1B 1C | ANL A, R6 1B 1C | XRL A, R6 1B 1C | MOV R6, #data 2B 1C | SUBB A, R6 1B 1C | MOV dir, R6 2B 2C | MOV R6, dir 2B 2C | CJNE R6, #data, rel 3B 2C | XCH A, R6 1B 1C | DJNZ R6, rel 2B 2C | MOV A, R6 1B 1C | MOV R6, A 1B 1C
F INC R7 1B 1C | DEC R7 1B 1C | ADD A, R7 1B 1C | ADDC A, R7 1B 1C | ORL A, R7 1B 1C | ANL A, R7 1B 1C | XRL A, R7 1B 1C | MOV R7, #data 2B 1C | SUBB A, R7 1B 1C | MOV dir, R7 2B 2C | MOV R7, dir 2B 2C | CJNE R7, #data, rel 3B 2C | XCH A, R7 1B 1C | DJNZ R7, rel 2B 2C | MOV A, R7 1B 1C | MOV R7, A 1B 1C